unroll

```
int H_out = H - K + 1;
int W_out = W - K + 1;
for (int b = 0; b < B; ++b)
for (int c = 0; c < C; ++c) {
int w_base = c (KK); for (int p
= 0; p < K; ++p)
for (int q = 0; q < K; ++q) {
for(in th= 0;h< H_out;++h)
for (int w = 0; w < W_out; ++w)
{
int w_unroll = w_base + p * K +
q;
int h_unroll = h * W_out + w;
X_unro ll[b, h_unroll, w_unroll]
= X[b, c, h + p, w + q];
```

Unroll

Unroll size C*K*K x H_out*W_out

Input (/ for rep) C*(H_out+K-1)*(W_out+K-1)

W forward

```
int m = blockIdx.x;
int h = blockIdx.y / W_grid +
thread Idx.y;
int w = blockIdx.y % W_grid +
thread Idx.x;
float acc = 0.;
for(in tc=0; c<C ;c++){
for(in tp= 0;p <K;p++) KxK
filter for (int q = 0; q < K;
q++)
                        acc +=
X[c, h + p, w + q] * W[m, c, p,
q];
}
       Y[m, h, w] = acc;
```

Convo matr

```
int X_tile_width = TILE_WIDTH +
K-1;
extern __shared__ float shmem[];
float* X_shared = &s hme m[0];
float W_shared = &s hme m[X -
_ti le_ width X_tile _wi dth];
m = blockI dx.x;
h_base = (block Idx.z / W_grid)
TILE_SIZE; the block w_base =
(block Idx.z % W_grid)
TILE_SIZE; x = thread Idx.x; ty
= thread Idx.y; h = h_base + tx;
w = w_base + ty;
float acc = 0.;
for (c = 0; c < C; c++)
tx and ty used as shorthand for
thread Idx.x and thread Idx.y
if (( ty < K) && ( tx < K))
W_shar ed[ty, tx]= W [m, c, ty,
tx];
__sync thr eads();
for (int i = h; i < h_base +
X_tile _width; i += TILE_W IDTH)
{ for (int j = w; j < w_base +
X_tile _width; j += TILE_W IDTH)
X_shared[i - h_base, j - w_base]
= X[n, c, i, j]}__s ync thr -
ead s() ;}Y[n, m, h, w] = acc;
```

Calculations

T/B 1024	warps 32 T
Warps 32 T	SM 8 B 1536 T

By **moutaz**
cheatography.com/moutaz/

Not published yet.
Last updated 12th December, 2017.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com